David M. Goldschlag. "Several Secure Store and Forward Devices," Proceedings of the Third ACM Conference on Computer and Communications Security, New Delhi, India, March, 1996, pages 129-137.

Several Secure Store and Forward Devices

David M. Goldschlag
Center For High Assurance Computer Systems
Naval Research Laboratory
Washington, D.C. 20375-5337
+1 202.767.2389
e-mail: goldschlag@itd.nrl.navy.mil

Abstract

DoD system high enclaves are often isolated from systems at other security levels because the usual connectors (guards) are expensive to procure, integrate, accredit, and operate, and usually require a human in the middle to review the data flow, independent of direction. This isolation reduces the effectiveness of information systems. The secure store and forward devices described in this paper can be used to solve an important (yet tractable) half of the problem: moving data from LOW to HIGH without a human in the middle. These devices were expressly designed to be easy to accredit. Security critical function is both minimized and separated from non-security critical function to reduce the need for trusted components. A prototype implementation of one of these store and forward devices is described as well.

Keywords: Accreditation, architecture, confidentiality, guards, high assurance, security, system engineering.

1 Introduction

System high operation is an effective means of providing mandatory access control (MAC). It assigns responsibility for MAC enforcement to both physical and personnel security, and requires no changes to the information system. But isolating systems reduces their effectiveness. Guards, which have traditionally been used to connect system high enclaves, are usually bidirectional. They require a human to monitor both downwards and upwards traffic, and are expensive to operate. This paper describes several secure store and forward devices that can be used to move information from LOW to HIGH. Since MAC policies do not restrict that sort of information flow, no human review is necessary.

Unfortunately, even a device designed to move information in one direction will often implicitly convey information in the opposite direction. For example, a memory write operation internal to a device can be delayed if the memory is locked. A cascade of these delays from the HIGH side of the device to its LOW side may enable a leak of information from HIGH to LOW. More obvious are delays imposed on LOW by HIGH's delays in consuming data. These sorts of return channels may be used as downward channels by modulating their content or speed (storage or timing channels, e.g., [5]). The challenge is to develop devices that allow sufficiently reliable communication without significant downward flow.

This paper describes several secure store and forward devices. The first is completely secure and has no trusted components, but is unreliable (i.e., data could be lost). The other devices are reliable extensions of the first device, and are constructed by adding simple trusted components that allow some downwards communication. The paper's focus, however, is on architectures that are easy to accredit. Accreditation is an approval process certifying that a device is not only believed to operate securely, but also that sufficient evidence exists to justify that belief, in the context of the operational requirement. For the purposes of accreditation, therefore, it is crucial to both minimize and isolate security critical function. Carefully designed architectures reduce the number and complexity of trusted components, thereby greatly reducing the cost of accreditation.

The basic approach to reducing complexity is to explore trade-offs between functionality and complexity. These trade-offs can be used to simplify the device or the environment's expectation of the device. For example, timing properties can be exploited: Placing strong timing constraints on the environment may simplify the device's operation. Although this trade-off makes the device more special purpose, a general purpose device may not always be preferable.

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **1996**		2. REPORT TYPE		3. DATES COVERED **00-00-1996 to 00-00-1996**	
4. TITLE AND SUBTITLE **Several Secure Store and Forward Devices**				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Research Laboratory,Center for High Assurance Computer Systems,4555 Overlook Avenue, SW,Washington,DC,20375**				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**9**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Separation also plays a key role: separating upwards channels from downwards channels, separating flow control (which coordinates the difference in rates between a producer and a consumer) from the acknowledgement of individual messages, and separating security critical components from non-security critical ones.

Also, identifying individual requirements facilitates the evaluation of alternative solutions for each requirement: Reliability may be obtained by acknowledgements or a reliable communications media; flow control may be obtained by modulating the timing of acknowledgements, by having a big buffer to handle bursts, or by placing timing constraints on the environment. Separating requirements, where possible, may permit solving them independently.

These considerations expand the design space and allow choices that reduce the amount of trusted hardware and software that needs to be analyzed carefully for correctness and reliability.

This paper is organized in the following way. Section 2 presents background information, including a description of the Pump[2, 3], an existing store and forward device for secure applications. Section 3 describes a one-way upwards channel whose security is essentially obvious. This channel is the foundation of all the store and forward devices presented later. Its implementation is also described. Section 4 presents several downgraders that can be used in conjunction with the upwards channel to provide acknowledgements for reliable communication. Section 4.4 describes a downgrader that provides Pump-like function for both reliable communication and flow control. Section 5 presents concluding remarks.

2 Background

MAC policies permit unrestricted information flow from LOW users to HIGH users. However, any information moved from HIGH to LOW must be reclassified or downgraded. This implies that in an electronic system, downward information flow includes a HIGH system's acknowledgement of data received from LOW. In this sort of secure environment, in the absence of downgrading, any device meant to move information from LOW to HIGH cannot directly relay acknowledgements from HIGH, and consequently cannot relay application level acknowledgements, containing error codes and other returned information. Therefore, it seems natural to characterize devices that securely pass data from LOW to HIGH as secure store and forward devices: they must accept data from LOW, acknowledge receipt of the data, and forward it on to HIGH. The acknowledgement returned to LOW is generated internally by the store and forward device and is unrelated to HIGH's eventual acknowledgement to the device. In applications where reliable communication is required, the device's acknowledgement is a guarantee that it will buffer the data until it is received by HIGH.

Consider a naive design of a store and forward device (figure 1):

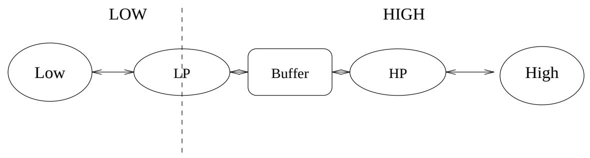

Figure 1: A Naive Design.

LOW sends data to LP (low process) which buffers it until HP (high process) forwards it to HIGH. As with any secure device, we must partition the store and forward device into LOW and HIGH components. Components which connect other LOW and HIGH components must be trusted. HP may be HIGH. The BUFFER, which contains HIGH information in the form of HIGH's response rate, is HIGH as well. LP, which then accesses both LOW and HIGH components, must be trusted to use this HIGH information without passing it along to LOW.

It would be convenient if this sort of cascading path did not exist. It therefore seems appropriate to partition LOW to HIGH communication into upwards and downwards channels. The upwards channel is the high capacity one; the downwards channel carries acknowledgements (or other error codes). It seems reasonable to build the upwards channel using standard components, to leverage off of existing networking technology. The downwards channel must be a trusted downgrader. For some simple subset of error codes (perhaps consisting only of ACKs) it is possible to construct an automatic downgrader that will filter out improper communication and consequently reduce the size of storage and timing channels, to make the HIGH to LOW channel capacity very small.

This architecture suggests two elements in a secure store and forward device: a completely one-way upwards channel, and a trusted downgrader that returns acknowledgements. The upwards channel cannot convey these acknowledgements, so it is impossible to make it reliable. It is important to note, however, that the reliable/unreliable distinction can only apply to an abstract communications protocol. For any implementation of a protocol, reliability is fundamentally a statistical property. A reliable protocol can lose data in a poor implementation; and, an unreliable protocol can be made more reliable by making components more robust.

The basic plan here is as follows: Develop a one-way upwards channel whose security is essentially obvious. This upwards channel provides unreliable communication (in principle), but in practice is very robust. This device is the foundation of the subsequent store and forward devices, and a prototype implementation is described here as well. On top of this foundation we add one of several trusted downgraders that convey acknowledgements from HIGH to LOW. The downgraders range from capacitor-like devices, to counters with timers, to devices that provide both reliable communication and flow control.

The next section (drawn largely from [1]) describes the basic Pump[1] as an example of a secure store and forward device. It serves as a good starting point for the subsequent discussion.

2.1 The Pump

The (NRL) Pump is a device that provides reliable communication from LOW to HIGH while limiting downward information flow. An abstract view of the Pump is shown in figure 2:

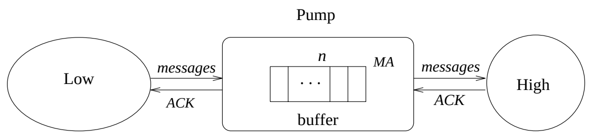

Figure 2: An Abstract view of the Pump.

The Pump places a non-volatile buffer (size n) between LOW and HIGH and sends ACKs to LOW at probabilistic times, based upon a moving average (MA) of the past m HIGH ACK times. A HIGH ACK time is the time from when the buffer sends a message to HIGH to the time when HIGH sends an ACK back. By passing ACKs to LOW at a rate related to HIGH's historical response rate, the Pump provides flow control and reliable delivery without directly conveying HIGH's response rate.

The timing of the ACKs from the Pump to LOW does represent a downward flow of information. However, the algorithm controlling the rate at which acknowledgements are returned is parameterized by the Pump's buffer size and the value of m, and the capacity of the timing channel from HIGH to LOW can be made as small as accreditors may require.

Because the Pump allows some downward communication, any operational implementation must be carefully analyzed to show that it indeed functions properly. One part of that evaluation has been done: a careful mathematical modeling of the Pump's specification permits the quantification of potential data leakage.

[1]The *network Pump,* which handles multiple senders and receivers in a networked environment, is described in[4].

Any implementation must also be evaluated to ensure that the specification was implemented correctly. At the level of detail presented in this section, it is impossible to identify which subsystems of the Pump must be trusted. A current implementation[7], running on a trusted machine, runs most subsystems at a privilege level that permits processes to violate the machine's security policy. Some of these subsystems do not need to violate the security policy (and indeed do not). Yet they are over-privileged, primarily because communication between processes running at different privilege levels is expensive. This design decision improves performance but makes accreditation more costly, since any subsystem that runs at a high privilege must be trusted.

The last store and forward device presented in this paper (see section 4.4) provides Pump-like function with fewer trusted components.

2.2 The Environment

A secure store and forward device is meant to move data from a LOW producer to a HIGH consumer. The behavior of these systems may be split into two classes. In the first class, over the long term, the consumer must be able to process data at least as quickly (on average) as the producer produces it. (This processing may include ignoring data that is too old.) In this case, it is sufficient to place a big buffer between the producer and the consumer that can moderate bursts[6]. (Determining the size of this buffer may be difficult[9].) In the other class, feedback from the consumer to the producer may be essential, because the relative rates of production and consumption are unknown, or the producer may depend on the consumer for flow control.

In many real systems, these two classes are mixed. For example, in database replication, replicate databases must, over the long term, be able to process updates quickly enough so the primary database is not delayed. So a big buffer is used to moderate bursts of activity on the primary. In normal operation, the big buffer prevents flow control from the replicate database from affecting the primary database: The big buffer fills or empties depending upon flow control from the replicate database, while the primary continues its asynchronous operation. If the big buffer fills (e.g., if the replicate database crashes), flow control then slows or stops the primary database.

The Pump provides flow control so the consumer can slow down the producer. Only the last store and forward device introduced in this paper provides flow control. The others may require that rare events like crashes be handled as special cases.

Even a store and forward device that provides flow control cannot simply be inserted between LOW and HIGH. Typically, the device will interrupt the normal communications protocol between two applications (by preventing application-level acknowledgements), so proxies for these applications must be built on each side of the device. The proxy on the LOW side functions as the HIGH application to the LOW application, and the proxy on the HIGH side functions as the LOW application to the HIGH application.

Flow control may simplify the overall design of a system. However, the benefits must be carefully considered in the context of the overall development and management life cycle of a system.

3 The Upwards Channel

This section describes a completely one-way upwards channel whose security is essentially obvious. This device is the foundation of all the subsequent store and forward devices, and a prototype implementation is described here as well.

The Upwards Channel has three key features:

- There is no downward channel. Communication from LOW to HIGH is completely asynchronous.
- The channel is composed of commercial networking products.
- The only trusted component has no control logic and stores no information, so evaluation is straightforward.

The Upwards Channel is unreliable because data may be lost at many points in transit, and HIGH cannot inform LOW that data was corrupted or lost during transmission. A functional view of the channel is depicted in figure 3:

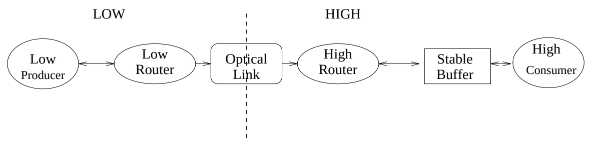

Figure 3: The Upwards Channel.

The OPTICAL LINK is a completely one-way communications media, described in the next section. The operation of the channel is as follows:

(1) LOW sends data to the LOW Router.

(2) The LOW Router packages the data and sends the packets over the OPTICAL LINK.

(3) The HIGH Router receives the packets and forwards them to the STABLE BUFFER.

(4) The STABLE BUFFER buffers the data until HIGH is ready to receive it.

For this channel to operate properly, the LOW Router and the OPTICAL LINK must be fast enough to process all data from LOW. The HIGH Router must be fast enough and the STABLE BUFFER must be both fast and large enough never to lose data from the OPTICAL LINK and still be able to forward data.

How large must the STABLE BUFFER be? The STABLE BUFFER must be large enough to buffer bursts of data from LOW until HIGH catches up. Many applications possess this sort of buffer already. Recall that in database replication, updates from the primary database may be buffered between the primary and the replicate databases to handle bursts of activity on the primary. That big buffer allows asynchronous operation of the primary and replicate databases, during normal operation. If the Upwards Channel is used to separate a LOW primary from a HIGH replicate database, the LOW primary will operate asynchronously, so the big buffer will function as the STABLE BUFFER and would be placed on the HIGH side (near the replicate database) in preference to the LOW side (near the primary).

Data could be corrupted or lost in transmission over the networks, in the routers, or over the OPTICAL LINK. The STABLE BUFFER could fail or overflow, especially if HIGH fails. Sending each packet multiple times or using forward error correction might reduce some of these losses, and losses could in any case be detected and flagged when uncorrupted data finally enters the STABLE BUFFER. Human intervention will be necessary to recover. It may be useful to make the STABLE BUFFER somewhat larger than it needs to be, and for the STABLE BUFFER to set off an alarm for its system administrator when the buffer is nearly full.

The lack of automatic recovery in the Upwards Channel is a consequence of the complete isolation of HIGH data from LOW. Because the physical design eliminates all downward information flow, the accreditation process is greatly simplified. The Upwards Channel is very much like the Big Buffer[6], except it has no trusted components.

3.1 A Prototype Implementation

Our prototype implementation of the Upwards Channel costs less than $7,000, and has only a small amount of custom (but untrusted) software. It uses two routers and the OPTICAL LINK. The OPTICAL LINK is a commercially available fiber-optic based device, already in use in many secure applications, that has only transmitters on the LOW side and receivers on the HIGH side. Communication over optical fiber requires a transmitter at one end and a receiver at the other to convert electronic digital signals to and from light. In a normal fiber-optic link, a single fiber can be used for bidirectional communication by having a transmitter and receiver pair at each end.

The OPTICAL LINK is a high speed (10 Mbps) and physically very reliable (data loss of better than 10^{-9}) serial connection. Although similar one-way function can be obtained by cutting a pin in a normal (electronic) serial connector, an optical solution is much more reliable, and provides less opportunity for electronic emissions, and is obviously one-way.

We use commercial routers to write to and read from the OPTICAL LINK. The LOW ROUTER has an Ethernet connection to the LOW network and a serial connection to the transmitting end of the OPTICAL LINK. The HIGH ROUTER has an Ethernet connection to the HIGH network and a serial connection to the receiving end of the OPTICAL LINK. There is no other connection between the two networks.

The OPTICAL LINK completely isolates HIGH data from LOW. The OPTICAL LINK is an asynchronous and one-way communications media, so LOW has no dependency on HIGH (not even memory write delays). This is apparent from the physical design of the OPTICAL LINK, and greatly simplifies accreditation.

UDP must be used for all communication intended to be forwarded across the OPTICAL LINK. TCP/IP cannot be used, because that protocol requires acknowledgements, which cannot be returned from the HIGH to the LOW network, because there is no data path from HIGH to LOW.

UDP is an unreliable protocol. But in practice, UDP is very reliable on a small Ethernet[10], and the OPTICAL LINK is very reliable. It is possible to decrease the chance of message loss and corruption by sending messages multiple times or using forward error correction. Of course, those mechanisms may not be effective if the network becomes disconnected, or if power goes down.

The prototype operates in the following way: The LOW producer sends UDP packets to the address of the STABLE BUFFER. We use strict source routing to route data through the LOW ROUTER to the OPTICAL LINK to the HIGH ROUTER to the STABLE BUFFER. The LOW ROUTER reads these UDP packets, encapsulates them in a form suitable for transmission over a serial line, and forwards them to the OPTICAL LINK which is attached to the router's serial port. The encapsulated packet is delivered to the HIGH ROUTER, which reconstructs the UDP packet and forwards it to the HIGH network. The STABLE BUFFER reads the packet, and buffers it until the HIGH consumer is ready to receive it.

Our application requires that data delivered to HIGH be delivered uncorrupted and in order. This implies that HIGH's view of LOW's data must be consistent with some history of the data that was sent. The prototype implementation of the UPWARDS CHANNEL therefore includes mechanisms for error detection and recovery. For error detection, the LOW producer adds checksums and sequence numbers to each packet, so the STABLE BUFFER can detect when a message is lost or corrupted and an error is signaled. The prototype does not attempt to reorder packets, so any out of order delivery is treated as a lost packet.

Recovery is handled in the following way: The LOW producer keeps a log of the packets that it has forwarded to the STABLE BUFFER. When the STABLE BUFFER signals an error (either a lost or corrupted packet), the error message includes the sequence number of the most recent packet it received successfully. The system administrator must then restart the LOW producer to resend all the subsequently (logged) packets.

How large must the LOW producer's recovery log be? It can be made arbitrarily small, if the LOW producer stops sending packets to the STABLE BUFFER once the log is full, and informs the system administrator when the log is about to fill. The system administrator must then determine the sequence number of a packet already successfully received by the STABLE BUFFER, and tell the LOW producer that all packets older than that sequence number need not be logged anymore.

This sort of error recovery requires a human system administrator, because there is no electronic channel over which the HIGH system may send control information to the LOW system. Because the Upwards Channel is quite reliable, overhead should not be high. All interventions by the system administrator should be logged, to help detect a malicious program's attempts to subvert the (human's) downward channel.

The custom software in this prototype implements both the recovery mechanism in the LOW producer (which runs on the LOW network) and the STABLE BUFFER (which runs on the HIGH network). This software is untrusted because compromise of this software will not compromise confidentiality.

3.2 Another Implementation Of The Upwards Channel

In the prototype described above, routers are used to move data between a network and the OPTICAL LINK. This simplifies the prototype considerably, because the router manages the packets and their conversion between UDP and encapsulated forms. Also, routers are simple and relatively inexpensive devices that can manage a high speed serial port.

Another implementation would limit the use of an unreliable protocol such as UDP to the portion of the communications path that must use an unreliable protocol. This requires surrounding the OPTICAL LINK with programmable devices such as workstations (figure 4):

The LOW producer and the HIGH consumer could

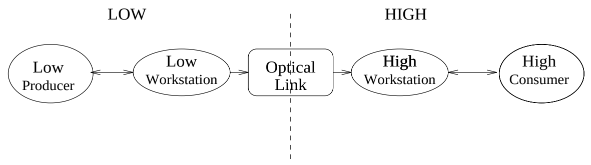

Figure 4: The Upwards Channel with Workstations.

use acknowledgement-based protocols to communicate with the LOW and HIGH workstations. Upon receipt of a packet, the LOW workstation acknowledges receipt to the LOW producer and forwards the packet over the OPTICAL LINK. The protocol then changes to an unreliable one without acknowledgements. The HIGH workstation reads packets from the OPTICAL LINK, buffers them in its STABLE BUFFER, and forwards them to the HIGH consumer using a reliable protocol.

The workstation implementation of the Upwards Channel limits the use of an unreliable protocol to the segment of the link that must use an unreliable protocol. This may be important in communication between larger enclaves because UDP over large networks may not be as reliable as over a small network.

Also, it is easy for the LOW workstation to predict when the HIGH workstation should get the packet it transmitted over the OPTICAL LINK. The timing analysis that this predictability permits is useful (see section 4.1) and may be more difficult to do in the router based solution.

3.3 Discussion

The Upwards Channel has no downwards communication; therefore it has no covert channels. Its reliability is a function of the reliability of the communications media. How well the STABLE BUFFER can manage flow control depends upon the application and the reliability of the consumer.

4 Downgraders for Reliability

In some applications, the Upwards Channel may be useful by itself, since it may be both sufficiently reliable and sufficiently easy to accredit. Our experiments will provide more data. Downgraders can be used in conjunction with the Upwards Channel in applications that require some feedback from HIGH to LOW, either to improve reliability or to provide flow control. The next sections present several downgraders that permit a HIGH consumer some control over the LOW producer. These provide a small downward channel whose added function may outweigh the risk of data leakage. The ACKs returned by any of these downgraders are still not application-level acknowledgements, however.

These downgraders are trusted devices that relay signals from HIGH to LOW. Even malicious manipulation of the input signals does not compromise security. This is crucial since the devices depend upon signals from untrusted processes on the system high enclaves.

These downgraders are most easily described in the context of the workstation implementation of the Upwards Channel (see section 3.2), which provides better control over data transmission time from LOW to HIGH.

4.1 A Downgrader for Acknowledgements

We use the behavior of a capacitor as a metaphor for the operation of this downgrader. The Capacitor provides acknowledgement of reliable communication over the OPTICAL LINK. It does not provide flow control from the HIGH consumer to the LOW producer, although it can be used to inform LOW that the STABLE BUFFER on the HIGH workstation has filled.

An untrusted connection from the HIGH workstation to the LOW workstation would violate our security requirements, because none of the code in either workstation can be trusted to protect data confidentiality. But the Capacitor is a very simple trusted downgrader between the two workstations that preserves confidentiality (figure 5):

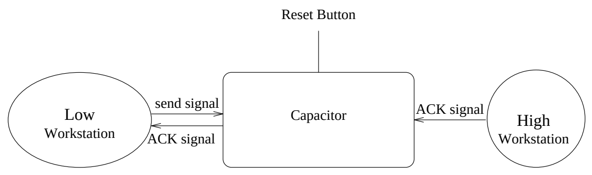

Figure 5: A Capacitor.

The fundamental idea here is to deliver ACKs back from the HIGH workstation at the end of a predefined interval, so the actual performance of the HIGH workstation is masked. This eliminates the covert timing channel. When the LOW workstation deposits data on the OPTICAL LINK, it also signals the Capacitor. This charges the Capacitor. If the Capacitor subsequently receives a signal from the HIGH workstation before it discharges, it will relay that signal to the LOW workstation when it discharges. If no signal arrives in time (or if it receives an unexpected signal), the Capacitor shuts down and sets off an alarm (which should be logged). No future signals will be relayed, until the device is reset. The Reset Button must not be connected to any untrusted system (e.g., it is pushed by the HIGH system administrator).

The discharge cycle must be long enough so the HIGH workstation can acknowledge receipt of data from the OPTICAL LINK nearly all of the time. Calculating the length of this cycle should be straightforward to do, because we know the data rate of the OPTICAL LINK, and the performance of the workstations.

In the normal case, acknowledgements will be passed back from HIGH to LOW. If there is a communications failure (the packet did not arrive or arrived in duplicate or out of order, or the STABLE BUFFER is full), or some malicious process is trying to manipulate the timing of acknowledgements, the Capacitor will shut down. The Capacitor provides acknowledgements but no automatic recovery.

4.1.1 Discussion

The Capacitor trades performance for the elimination of the implicit timing channel that accompanies an acknowledgement from HIGH to LOW. All acknowledgements are returned after the worst case communications delay has elapsed. This is akin to time slicing with fixed time slices between processes (whether the process needs the time or not) to satisfy non-interference requirements.

In the workstation version of the Upwards Channel, however, the performance hit is very low. This is because the worst case communications delay should be very close to the average communications delay. If, in the context of the larger environment, even this delay is unacceptable, one could always increase the speed of the LOW and HIGH workstations and the OPTICAL LINK so their worst case behavior satisfies the timing requirements.

The Capacitor does not depend upon the correct operation of the HIGH and LOW workstations to maintain security. Malicious behavior will close the downward channel.

The Capacitor introduces a covert channel from HIGH to LOW. The penalty for using this covert channel is very high, however: the downward channel closes, and an alarm is set off (which can be logged). The capacity of this covert channel is highest when the Upwards Channel is reliable. For example, in one scheme for HIGH to send LOW a message coded as the number V, LOW should send a stream of packets, and HIGH should fail to acknowledge the $V+1$'th packet. On average HIGH can send LOW a message of (at most) N bits if LOW sends HIGH 2^{N-1} packets. In general, the communications cost is exponential, and the channel subsequently shuts down.

A cheaper use of this channel uses a guessing game. Imagine that LOW has a good guess at the N bits that HIGH wants to send. So LOW sends each bit, and the first bit that is unacknowledged means that LOW guessed wrong. Although the capacitor then shuts down, HIGH has successfully passed to LOW one additional bit, in addition to confirming the earlier guessed bits. The communications cost is linear with the number of bits.

These covert channels confirm the observation that once a downward channel is present, it is virtually impossible to prevent the leakage of small messages. This observation was aptly named the *Small Message Criterion* in [8].

The design of the Capacitor is an example of placing strong timing constraints on the environment (the LOW and HIGH workstations) in order to simplify the trusted device. These constraints could complicate the management of the larger system.

4.2 Extensions of the Capacitor

It is easy to imagine simple extensions of the Capacitor that add some amount of recoverability at the risk of increased data leakage. One extension includes a counter that permits a specified number of failures to occur before the device shuts down. Another option is to include both a counter and a timer to limit how often automatic recovery could happen, and how long a penalty period should be. Failures should still set off alarms and be logged.

4.3 A Heartbeat

Capacitor-like downgraders do not distinguish between communications failure due to message corruption or loss, and machine failure. If we install a trusted Heartbeat Relay from the HIGH workstation to the LOW workstation, the LOW workstation can interpret the lack of a heartbeat as a sign that the HIGH workstation is down. As in the Capacitor, the Heartbeat Relay would relay signals from the HIGH workstation at the end of each discharge cycle, and then automatically recharge. If the HIGH workstation missed a signal, the Heartbeat Relay shuts down. A Heartbeat Relay can also be extended with counters and timers.

4.4 A Downgrader for Flow Control

One can also obtain Pump-like flow control by means of a downgrader. Like the Capacitor, the Flow Control downgrader expects an acknowledgement from the HIGH workstation that the most recently sent packet has been received. However, this acknowledgement is not immediately relayed to the LOW workstation. Instead, the acknowledgement is delayed based upon the HIGH consumer's historical average response time.

The Pump computes its moving average by maintaining a list of the previous m HIGH ACK times (see section 2.1) and averaging those values. The Flow Control downgrader could maintain that list also. If we want to restrict the downgrader's memory to simple counters, an approximation to the moving average can be calculated by folding new HIGH ACK times into the historical average of HIGH response rates, by weighting the historical average appropriately (e.g., compute the new historical average by multiplying the previous historical average by $m-1$, adding in the new HIGH ACK time, and dividing by m).

HIGH ACK times are determined using two additional input signals on the Flow Control downgrader (figure 6). One is used each time the HIGH workstation forwards data from its STABLE BUFFER to the HIGH consumer; the other is used each time the HIGH consumer acknowledges receipt of that data. The downgrader notes the time that elapses between signals and computes the new historical average.

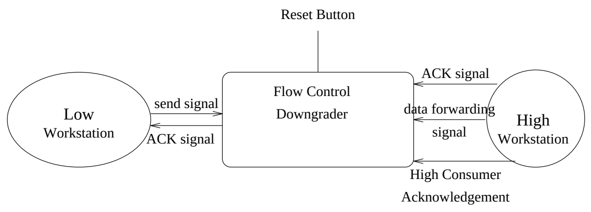

Figure 6: The Flow Control Downgrader.

The behavior of the Flow Control downgrader is very similar to the Capacitor's. When the LOW workstation deposits data on the OPTICAL LINK, it also signals the Flow Control downgrader, defining the beginning of the acknowledgement interval. The HIGH workstation must acknowledge receipt of the data from the OPTICAL LINK by the end of that interval. But instead of simply relaying the signal at the end of that interval, the Flow Control downgrader delays a additional amount corresponding to the current historical average response time.

What happens if data is lost over the OPTICAL LINK or if the STABLE BUFFER fills? Neither should happen often. But we are then faced with the same situation the Capacitor can be faced with: Either the downgrader shuts down and must be reset, or a combination of counters and timers, along with a recovery penalty period may allow for automatic recovery. In either case an alarm should be set off and logged.

4.4.1 Discussion

The Flow Control downgrader provides flow control because delays in the HIGH consumer's response rate are passed to LOW, in the form of small fluctuations in the delay of acknowledgements: these fluctuations persist through many acknowledgements. A longer historical average makes the fluctuations smoother and more persistant, reducing the size of the covert channel.

It is interesting to contrast the Pump with the store and forward device which is a combination of the Upwards Channel and the Flow Control downgrader (call this the Pump-like device).

In the Pump-like device, an acknowledgement may be returned at or after the predefined interval corresponding to the Capacitor's discharge cycle. This interval is required because the HIGH workstation is untrusted and may modulate its delay when delivering the acknowledgement. The Pump defines no similar interval. It is likely, however, that in an implementation of the Pump such an interval implicitly exists; otherwise LOW could infer the load on the trusted system implementing the Pump, and this load may be related to HIGH's operation.

The Pump-like device must treat a full buffer as a special case, requiring some recovery procedure. In the Pump, communication continues, but a covert channel opens.

If data is lost, the Pump-like device again depends upon some recovery procedure. The Pump, however, will simply not acknowledge lost or corrupted data, and LOW will resend the lost data after a timeout period. This difference, again, is because Pump subsystems are trusted to acknowledge data properly, while the HIGH workstation cannot be trusted.

5 Conclusion

This paper describes several secure store and forward devices that may be used to move data from a LOW to a HIGH enclave. These systems are designed to be easy to accredit by limiting the complexity of trusted components.

The design principle used here partitions the upwards channel from the downwards channel in order to isolate security critical function. This permits the use of appropriate off-the-shelf networking equipment for the Upwards Channel. The security of the Upwards Channel is essentially obvious because it uses an asynchronous and one-way communications media (the OPTICAL LINK) that completely isolates HIGH data from LOW.

Downward channels are provided via downgraders; these are the only trusted components. The downgraders place a timing constraint on the environment to eliminate timing channels: data must be acknowledged within the expected time. The acknowledgement is then relayed at the expected time. Although this is a strong timing constraint, it does not extend past the systems that directly write to or read from the OPTICAL LINK, so its impact is localized.

The Flow Control downgrader passes flow control information from HIGH to LOW, but, like the Pump, reduces the size of the covert timing channel by using a historical average of acknowledgement delays instead of passing delays directly. The combination of the Flow Control downgrader and the Upwards Channel provides a Pump-like communications device. The Pump-like device and the Pump behave differently when data is lost over the OPTICAL LINK or when the STABLE BUFFER fills, although these differences may be reduced by permitting a limited amount of automatic recovery in the Flow Control downgrader. The differences are direct consequences of limiting the trusted components to the downgrader.

The store and forward devices presented here provide a range of function suitable for a variety of environments. In some cases, the Upwards Channel may be sufficient by itself. Other cases may also require flow control. The benefits of a more complex downgrader must be carefully weighed against their ultimate impact on the life cycle of the system.

6 Acknowledgements

Discussions with many people improved this paper. Andy Moore and John McDermott helped develop the basic ideas; Bruce Montrose proposed the trusted Heartbeat Relay. Carl Landwehr suggested using a capacitor as a metaphor for the downgrader. Paul Syverson proposed the first attack on the covert channel in the Capacitor. Steven Greenwald, Cathy Meadows, Bruce Montrose, Andy Moore, and Ira Moskowitz read the paper carefully, as did the anonymous referees.

References

[1] J. N. Froscher, D. M. Goldschlag, M. H. Kang, C. E. Landwehr, A. Moore, I. S. Moskowitz, C. N. Payne. "Improving Inter-Enclave Information Flow for a Secure Strike Planning Application," Proceedings of the 11'th Annual Computer Security Applications Conference, New Orleans, Louisiana, December, 1995.

[2] M. H. Kang and I. S. Moskowitz. "A Pump for rapid, reliable, secure communication," Proceedings ACM Conf. Computer & Commun. Security '93, pp. 119 - 129, Fairfax, VA, 1993.

[3] M. H. Kang and I. S. Moskowitz. "A data Pump for communication," NRL Memo Report 5540-95-7771.

[4] M. H. Kang, I. S. Moskowitz and D. Lee."A network version of the Pump," Proc. of the IEEE Symposium on Research in Security and Privacy, pp. 144 - 154, Oakland, CA, May 1995.

[5] B. Lampson. "A Note on the Confinement Problem," *Communications of the ACM*, Vol. 16, No. 10, 1973.

[6] J. McDermott, "The b^2/c^3 problem: how big buffers overcome covert channel cynicism in trusted database systems," in *Database Security VIII, Status and Prospects, J.Biskup, M. Morgenstern, C. Landwehr, eds.*, IFIP Transactions A-60, Elsevier Science B.V., Amsterdam, ISBN 0 444 81972 2, pp.111-122.

[7] B. E. Montrose and M. H. Kang. "An Implementation of the Pump: Event Driven Pump," NRL Memo. Report 5540-95-7782, 1995.

[8] I. S. Moskowitz and M. H. Kang. "Covert channels — Here to stay?," Proceedings COMPASS '94, pp. 235 - 243, Gaithersburg, MD, 1994.

[9] I. S. Moskowitz and M. H. Kang. "The modulated input modulated output model," To appear: *Database Security 9, Status and Prospects*, IFIP Transactions, Elsevier Science B.V., Amsterdam. In the interim is available in the preliminary conference proceedings of the 9th annual IFIP WG 11.3 Working Conference on Database Security, August 1995.

[10] W. R. Stevens. *Unix Network Programming,* Prentice Hall Software Series, 1990.